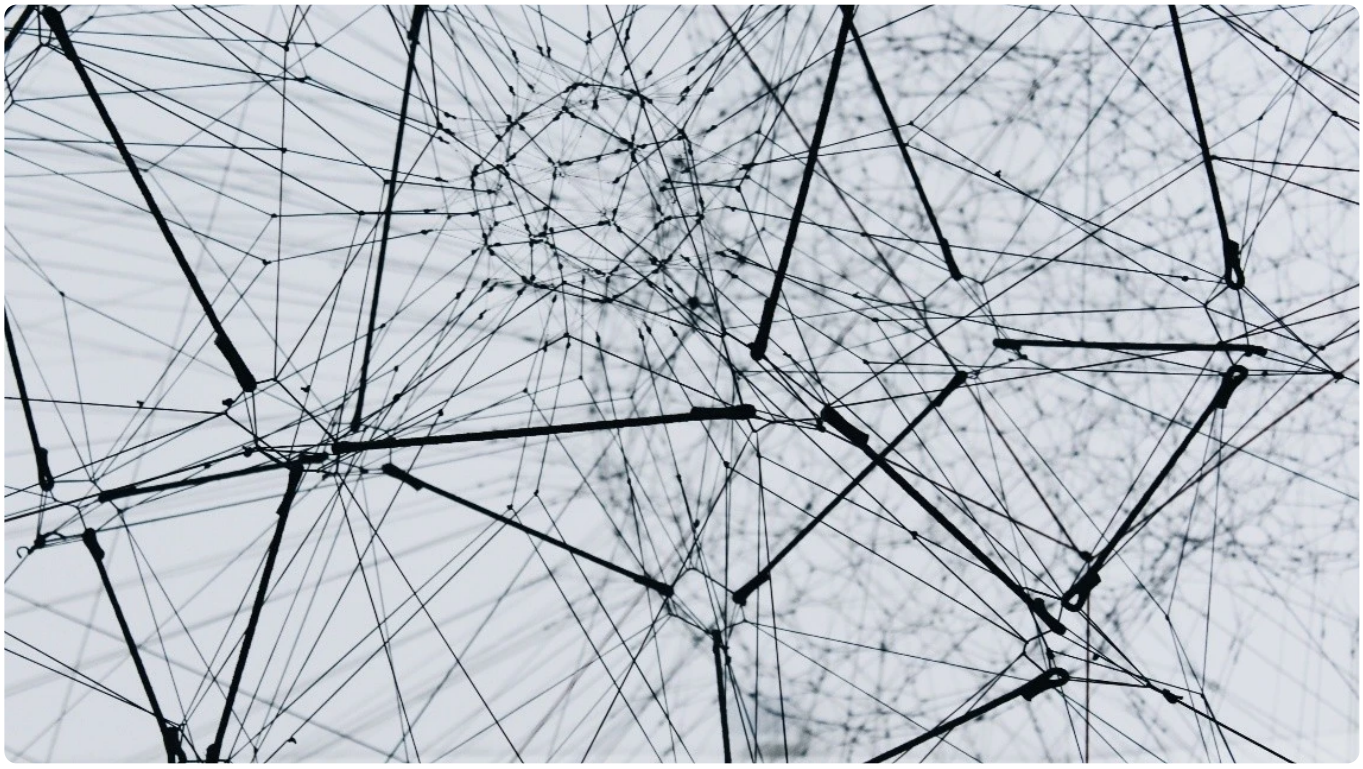
下载APP

02 | 学习方法：建立你的大数据知识网络

2021-09-22 徐文浩

《大数据经典论文解读》 课程介绍 >



讲述：徐文浩

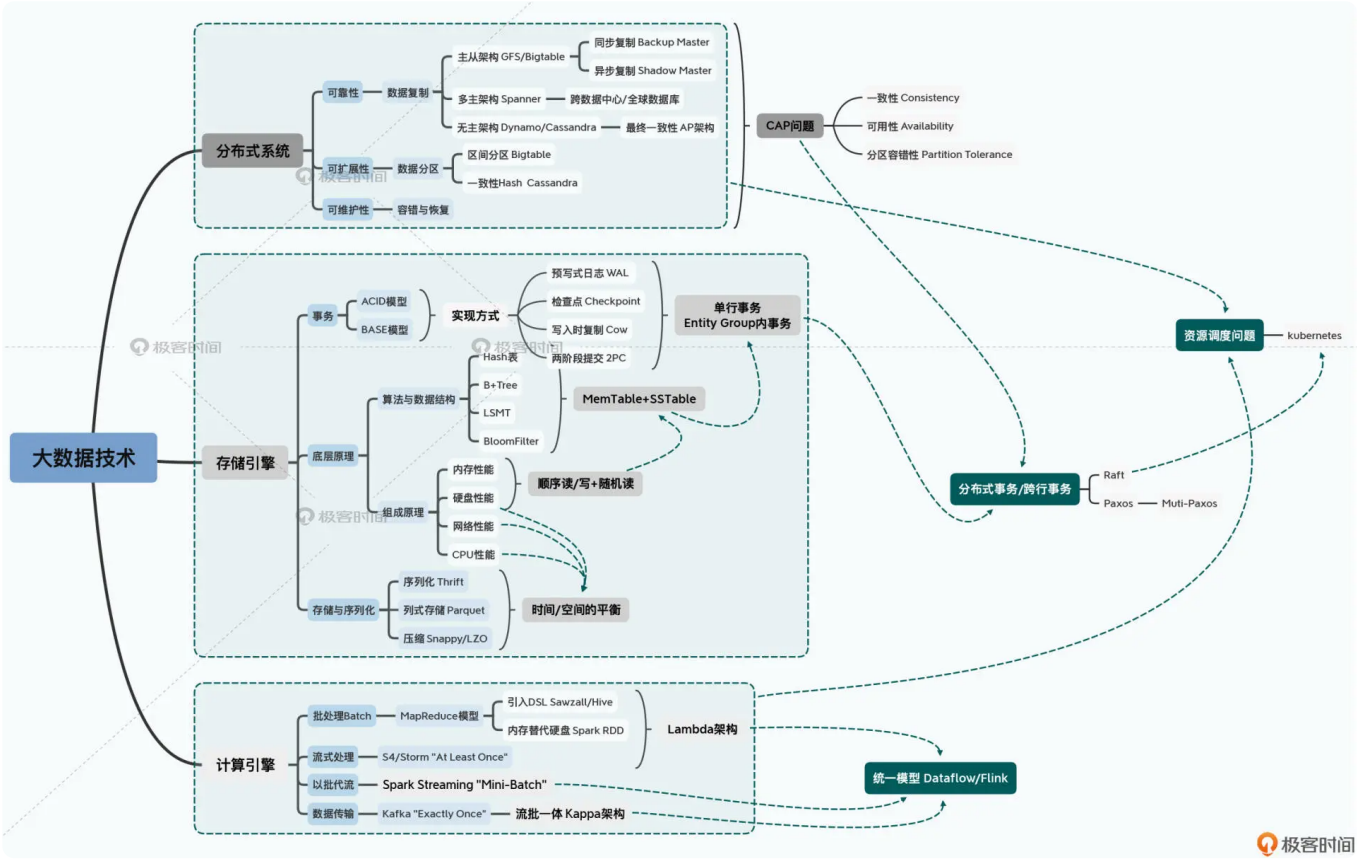
时长 15:47 大小 14.46M

你好，我是徐文浩。

在了解了大数据论文之间的脉络后，接下来，我们就要进入精读论文的学习当中了。不过，在具体解读一篇篇的论文之前，我想先带你来一起看一看，这些大数据论文到底涵盖了哪些知识点，这些知识点又是来自于大数据系统中的哪一个组件。通过梳理这些组件涵盖了什么知识点，你就能更好地理解和掌握大数据领域相关的知识全貌。

毕竟，相比于某一门计算机课程、某一门编程语言或者某一个开源框架，“大数据”涉及到的知识点多而繁杂。所以这里，我就整理了一份知识地图，好让你对课程所涉及到的知识点有迹可循。

大数据领域的知识地图



大数据技术知识点

从这张图可以看出，要想了解和学习“大数据”领域的相关知识，我们可以从三个维度来切入。

分布式系统

所有的大数据系统都是分布式系统。我们需要大数据系统，就是因为普通的单机已经无法满足我们期望的性能了。那么作为一个分布式的数据系统，它就需要满足三个特性，也就是可靠性、可扩展性和可维护性。

第一个，作为一个数据系统，我们需要**可靠性**。如果只记录一份数据，那么当硬件故障的时候就会遇到丢数据的问题，所以我们需要对数据做复制。而**数据复制**之后，以哪一份数据为准，又给我们带来了主从架构、多主架构以及无主架构的选择。

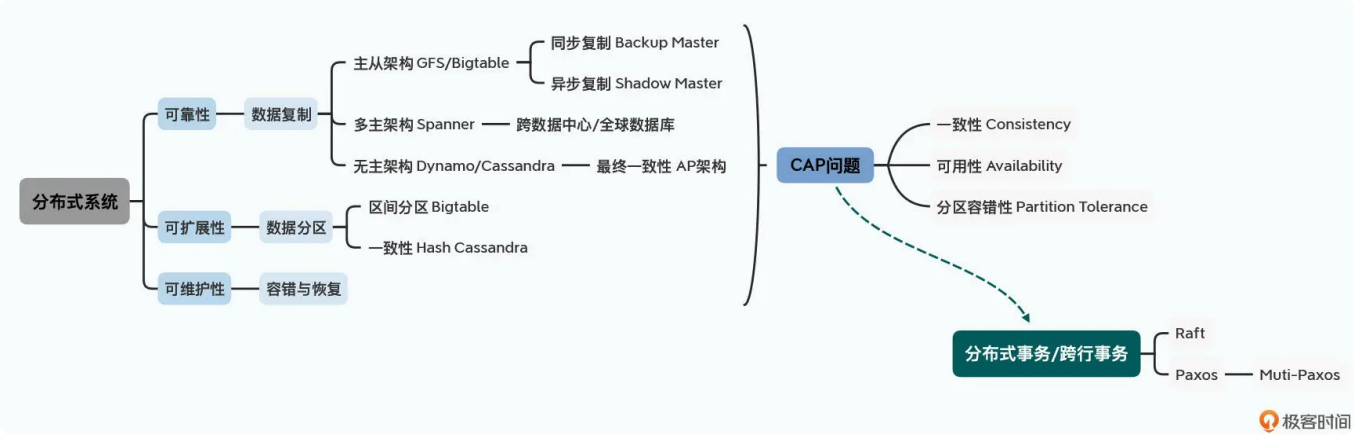
然后，在最常见的主从架构里，我们根据复制过程，可以有同步复制和异步复制之分。同步复制的节点可以作为高可用切换的 Backup Master，而异步复制的节点只适合作为只读的 Shadow Master。

第二个重要的特性是**可扩展性**。在“大数据”的场景下，单个节点存不下所有数据，于是就有了**数据分区**。常见的分区方式有两种，第一种是通过区间进行分片，典型的代表就是

Bigtable，第二种是通过哈希进行分区，在大型分布式系统中常用的是一致性 Hash，典型的代表是 Cassandra。

最后一点就是整个系统的**可维护性**。我们需要考虑容错，在硬件出现故障的时候系统仍然能够运作。我们还需要考虑恢复，也就是当系统出现故障的时候，仍能快速恢复到可以使用的状态。而为了确保我们不会因为部分网络的中断导致作出错误的判断，我们就需要利用**共识算法**，来确保系统中能够对哪个节点正在正常服务作出判断。这也就引出了 CAP 这个所谓的“不可能三角”。

而分布式系统的核心问题就是 CAP 这个不可能三角，我们需要在一致性、可用性和分区容错性之间做权衡和选择。因此，我们选择的主从架构、复制策略、分片策略，以及容错和恢复方案，都是根据我们实际的应用场景下对于 CAP 进行的权衡和选择。



单节点的存储引擎

然而，即使是上万台的分布式集群，最终还是要落到每一台单个服务器上完成数据的读写。那么在存储引擎上，关键的技术点主要包括三个部分。

第一个是**事务**。在写入数据的时候，我们需要保障写入的数据是原子的、完整的。在传统的数据库领域，我们有 **ACID** 这样的事务特性，也就是原子性（Atomic）、一致性（Consistency）、隔离性（Isolation）以及持久性（Durability）。而在大数据领域，很多时候因为分布式的存在，我们常常会退化到一个叫做 **BASE** 的模型。BASE 代表着基本可用（Basically Available）、软状态（Soft State）以及最终一致性（Eventually Consistent）。

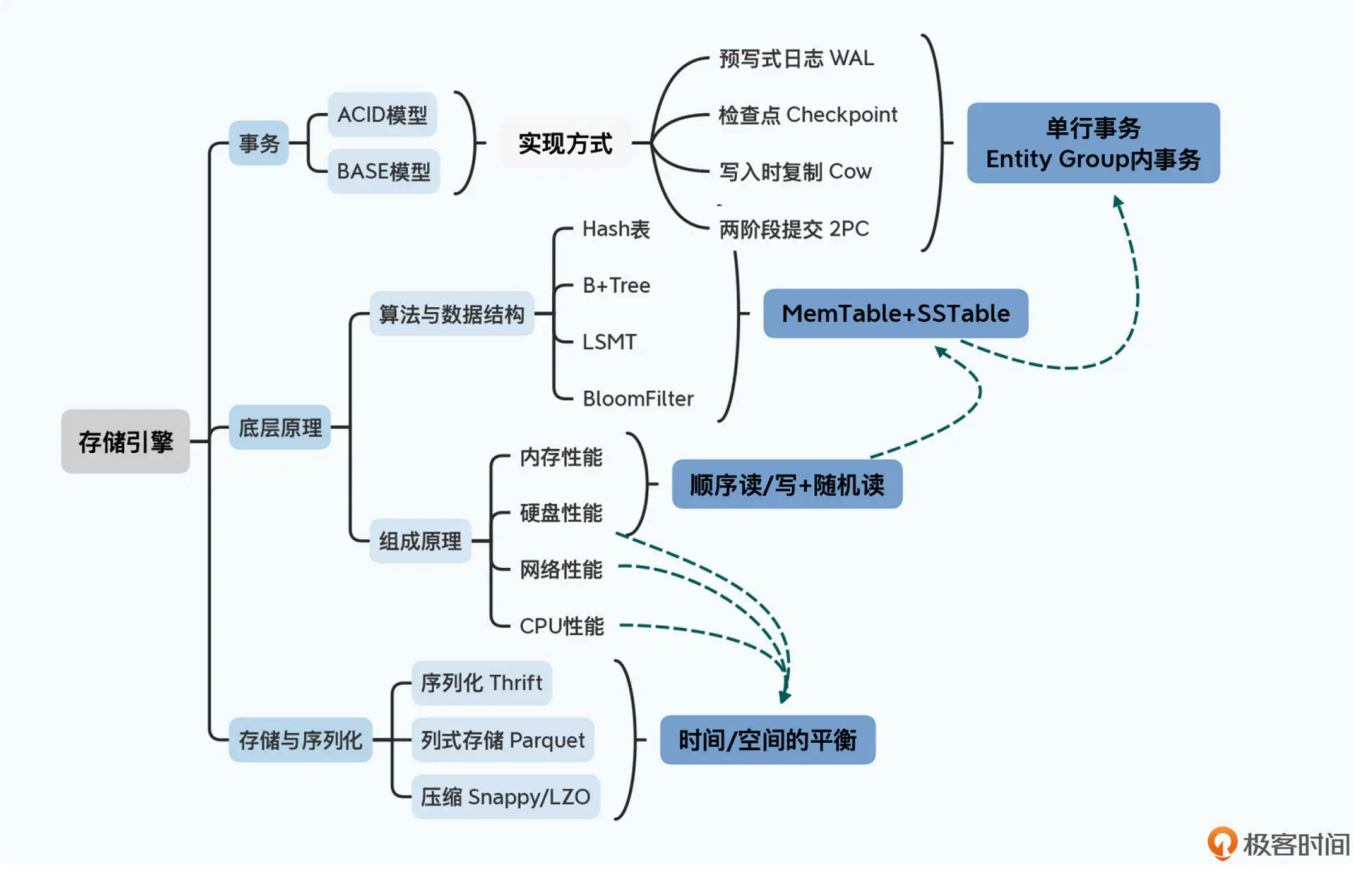
不过无论是 ACID 还是 BASE，在单机上，我们都会使用预写日志（WAL）、快照（Snapshot）和检查点（Checkpoints）以及写时复制（Copy-on-Write）这些技术，来

保障数据在单个节点的写入是原子的。而只要写入的数据记录是在单个分片上，我们就可以保障数据写入的事务性，所以我们很容易可以做到单行事务，或者是进一步的实体组（Entity Group）层面的事务。

第二个是底层的数据是**如何写入和存储**的。这个既要考虑到计算机硬件的特性，比如数据的顺序读写比随机读写快，在内存上读写比硬盘上快；也要考虑到我们在算法和数据结构中的时空复杂度，比如 Hash 表的时间复杂度是 O(1)，B+ 树的时间复杂度是 O(logN)。

这样，通过结合硬件性能、数据结构和算法特性，我们会看到分布式数据库最常使用的，其实是基于 LSM 树（Log-Structured Merge Tree）的 MemTable+SSTable 的解决方案。

第三个则是**数据的序列化**问题。出于存储空间和兼容性的考虑，我们会选用 Thrift 这样的二进制序列化方案。而为了在分析数据的时候尽量减少硬盘吞吐量，我们则要研究 Parquet 或者 ORCFile 这样的列存储格式。然后，为了在 CPU、网络和硬盘的使用上取得平衡，我们又会选择 Snappy 或者 LZO 这样的快速压缩算法。



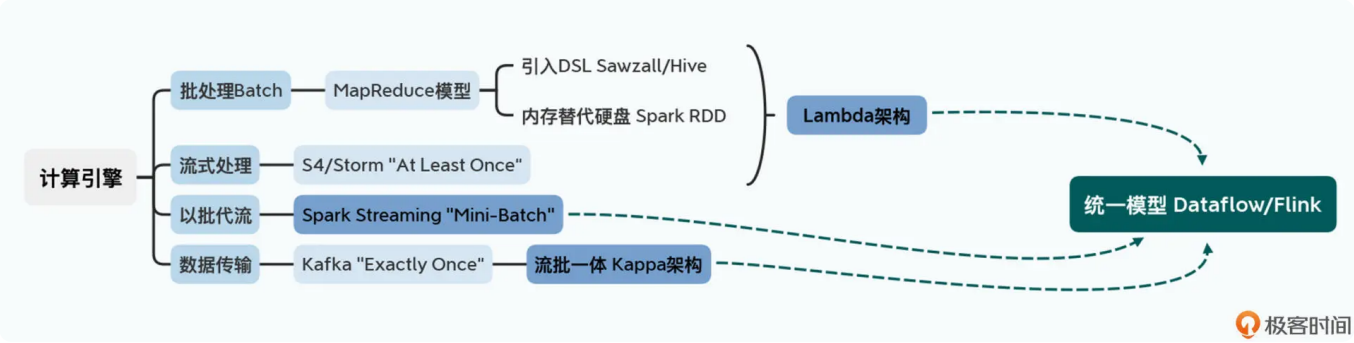
计算引擎

这个维度实际上也是大数据领域本身进化和迭代最快的一部分。为什么会这么说呢？让我们来一起捋一下大数据处理引擎的进化过程：

> 我们先有了最原始粗糙的 MapReduce 来进行**批数据处理**，然后围绕它不断迭代出了让数据处理更快的 Spark 和让数据处理更容易的各种 DSL（比如 Sawzall/Pig 和 Hive）。
>
> 然后我们围绕着**实时数据处理**，有了“最少一次”的 S4/Storm，并把它和批处理综合到一起，产生了著名的 Lambda 架构。
>
> 紧接着有了“**以批为流**”，通过 Mini-Batch 来进行实时数据处理的 Spark Streaming，以及“**流批一体**”，能够做到“正好一次”的 Kafka 和 Kappa 结构。
>
> 最后，还是 Google 一锤定音，给出了统一的 Dataflow 模型，并伴随着有了 Apache Flink 和 Apache Beam 这两个开源项目。

分布式问题，往往脱胎于少量经典论文的算法证明；单节点的存储引擎，也是一个自计算机诞生起就被反复研究的问题，这两者其实往往是经典论文的再现。但是在上千个服务器上的计算引擎应该怎么做，则是一个巨大的工程实践问题，我们没有太多可以借鉴的经验。这也是为什么计算引擎的迭代和变化是最大的。

不过随着 Dataflow 论文的发表，我们可以看到整个大数据的处理引擎，逐渐收敛成了一个统一的模型，大数据领域发展也算有了一个里程碑。



调度系统和综合应用

总结来说，分布式系统、存储引擎和计算引擎就共同构成了大数据的核心技术。更进一步，随着多种分布式系统的混排，又产生了 Kubernetes 这样的资源管理和调度系统。而所有的这些技术之间，都不是各自独立，而是相互关联的。

大数据技术其实是计算机科学中很多科目的综合应用。在上面的知识地图里，我们可以看到在单节点上的存储引擎，就是要综合考虑组成原理、算法和数据结构以及数据库原理相关的知识。而序列化和压缩，前者是组成原理里的二进制编码问题，后者则脱胎于算法和数据结构中的赫夫曼树和赫夫曼编码。

另外，最终选择什么算法做压缩，又要回到组成原理中，对于 CPU、网络以及硬盘的硬件性能进行平衡和考量。而针对分布式事务，我们一方面需要理解单机下的数据库事务，另一方面需要理解分布式环境下的 CAP 不可能三角。只有这样，我们才能对于 Paxos 以及 Raft 这些共识算法有深入的理解。

而当我们要优化海量数据的分析效率，需要修改的反而是单节点存储引擎，因为只有通过列式存储，我们才能优化海量数据分析中的瓶颈：读取硬盘数据的 IO。

因此，从我的认知来看，大数据系统的知识点不是一棵树，而是一张网。当你学明白了整个大数据系统的知识点和原理之后，自然就有了深厚的计算机科学和工程的功底。它能给你一种，“天下虽大，何处去不得”的信心。

精读论文学知识，诀窍在哪里？

不过，即使有了这张大数据知识地图之后，你或许还会遇到一些难题，比如说，面对这些相对分散和全面的知识点，是学习大数据论文的第一层挑战。而论文本身往往也很精炼，则是学习过程中的第二层挑战。

所以为了帮助你更好地学习和精读论文，我为你总结了三个学习方法。

首先，是从第一性原理出发，尝试自己去设计系统和解决问题。

一篇篇的大数据论文，并不是教科书里的一个章节或者一个知识点，而是对于一个重要的系统问题的解决方案。在读论文之前，先尝试自己去思考和解决对应的问题，有助于你更深刻地理解问题和解决方案的重点。

比如，在学习 Megastore 的论文之前，你可以问一问自己下面这两个问题：

如果要在 Bigtable 上为数据表加上 Schema，我该怎么做？

如果我希望能够在 Bigtable 上，支持跨行事务，可以从哪里起步？

无论你自己的思考和答案是否正确，带着你对问题的思考和方案去读论文，你的收获一定比囫囵吞枣地读一遍要多得多。

其次，是多做交叉阅读和扩展阅读。

论文本身往往只有 10 来页，非常精炼，对于很多知识点，往往就只有一个小片段，甚至只有一两句话，所以交叉阅读和扩展阅读少不了。根据你需要深入了解的知识点，你可能要回顾之前已经解读过的论文，也可能需要去阅读一些开源项目的代码，或者是一些计算机经典书籍中相关的章节，帮你彻底理解对应的问题。

比如，学习 Bigtable 论文的时候，论文里只告诉你底层的数据存储是 SSTable。而通过学习 LSM 树，或者是去读一下 LevelDB 的源码，你不仅可以理解 SSTable 的底层实现。还能帮助你深入理解针对硬件性能去设计数据结构，乃至系统中特定的组件。当然，我在课程的讲解中，也会给你推荐一系列的扩展阅读资料，帮助你找到更多的学习线索。

最后，是给自己制定一个明确的学习目标，然后围绕学习目标，进行泛读和精读、理论和实践的结合。

大数据的论文，是一个一个“点”，但是每一个点深挖下去，都可以串联到大量的理论和工程知识。而如果通过阅读论文追求学得“多”，其实意义并不大。最合适的学习方法，我认为是有针对性地针对自己的目标，来学习这个课程。

如果你的工作就是开发和维护大数据系统中的某个项目，比如 HBase、Flink，那么你就精读对应的论文，泛读其他的相关论文，并对于你所关心的项目源码进行深入挖掘。搞清楚每一个设计背后选择的根本原因，搞清楚它为什么这么设计。

如果你原先是做后端应用开发，想要学习大数据知识，转向大数据领域的开发。那么，搞清楚每篇论文和每个系统的应用场景，尝试通过 Google Cloud 或者其他的云系统，多尝试用一用这些大数据系统，会更有帮助。

如果你就是想要提升自己的理论知识和架构能力，那么我建议你放慢节奏。搞清楚论文里每一个关键设计点的原理，尽量多阅读我给到的推荐阅读材料。甚至你不妨可以动手试一

试，去实现其中的一些算法和组件，这是最有效的办法。

其实，归根到底，学习的成效不是表面知识知道得多，而是要真的掌握和理解这些知识。AngelList 的创始人纳瓦尔·拉威康（Naval Ravikant）说过这样一段话，听了之后让人觉得“与我心有戚戚焉”。

他说，“知识是一座摩天大楼。你可以在记忆的脆弱基础上走捷径，或者在理解的钢架上慢慢建立。”希望你在后面的学习过程中，像他说的一样，能够真正理解论文里提出的问题和解决方案，构建起你自己的大数据摩天大楼。

给你一些“参考文献”

学习还是要花笨功夫的。在课程之外，多阅读、多搜索、多看不同来源的资料，多和朋友、同事、老师一起交流，一定能够帮你掌握好要学习的知识点。

在整个大数据领域，也有许多经典图书和论文。我在每一讲里，也都会留下一些“推荐阅读”的资料。在这里，我也给你推荐一些我自己读过、看过的经典书籍和课程。如果你想深入理解大数据技术，读一读这些内容，绝不会让你后悔的。

首先是 Storm 的作者南森·马茨（Nathan Marz）的“Big Data”，现在也有中译本叫做《大数据系统构建》。对于人为错误的容错问题的思考，为我们带来了著名的 Lambda 架构。在我看来，即使到今天 Lambda 架构也并不过时。

其次是俗称 DDIA 的这本《数据密集型应用系统设计》，这本书梳理了整个大数据领域的核心技术脉络，是一本非常合适的架构入门书。

第三本是专注于流式处理的《Streaming System》，不过目前还没有中译本上市。

如果你更喜欢通过视频课程学习，那么去看一看来自 MIT 的课程 6.824 的 Distributed System 绝对错不了。我在这里放上了Youtube和B 站的视频链接。

最后是一份很容易被人忽视的资料，就是 2009 年 Jeff Dean 在 Cornell 大学的一个讲座“Designs， Lessons and Advice from Building Large Distributed Systems”的 PPT，我也推荐你去看一看，对于理解大数据系统的真实应用场景很有帮助。

课后思考

那么，今天这一讲就到这里了。在这一讲的知识地图里，你觉得还可以补上一些什么知识点？在你了解的大数据系统和知识点中，你还能找到什么联系？

欢迎你在留言区写下你的学习目标和学习计划，和大家一起交流。也欢迎你把今天的内容分享给你的朋友，互相督促，共同成长。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 5 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 01 | 什么是大数据：从GFS到Dataflow，12年大数据生态演化图

下一篇 03 | The Google File System（一）： Master的三个身份

精选留言 (9)

写留言

冯磊

2021-09-22

数据密集型系统设计，这本书在刚进入大数据领域时简单看过，带着很多疑问做项目。三

年下来，发现很多点慢慢串成了线，线变成了网，到如今这本书已经读了五遍。从结构化，非结构化到半结构化数据系统的演变，慢慢都在趋向异构数据系统的设计。大数据世界越来越星光璀璨，所以从底层建立起一套知识体系太重要了。

展开

5

lei Shi

2021-09-22

ddia是真的顶，现在还没梳理明白，继续学习

展开

3

pedro

2021-09-22

读论文一定要画脑图啊，一个点一个点的串起来，纵观横缆以后，才能感受到一个领域的自洽性。

2

bymzy

2021-09-22

参考网上大神的代码，实现过mit 6.824 2012的大项目，还是非常有收获的

1

dog_brother

2021-09-23

知识密度好大的一章啊，这其中只有50%的知识点是听过的，需要好好补课啦

yeyuliunian

2021-09-22

想问老师一个问题：单行事务和跨行事务的区别？
从文中看好像是指是否跨分片，这里的的“行”如何理解呢？

展开

zhanyd

2021-09-22

MIT 的课程 6.824 的 Distributed System 这视频我看过几课，就是作业太难了。老师在课上讲了一些概念，然后作业就要完成一个小项目，然后我就不会了。。。

在路上

2021-09-22

徐老师好，经过老师的梳理，对大数据整体的知识结构清晰了很多，非常感谢老师荐书，之前自己找书，虽然学了不少，但不知道哪一本才是经典。

展开

闻人

2021-09-22

网络通信方式：rpc